Evolving Robust Supervisors for Robot Swarms in Uncertain Complex Environments

Elliott Hogg[1], David Harvey, Sabine Hauert[1], and Arthur Richards[1]

Bristol Robotics Laboratory, University of Bristol, Bristol UK
elliott.hogg@bristol.ac.uk

Abstract. Whilst swarms have potential in a range of applications, in practical real-world situations, we need easy ways to supervise and change the behaviour of swarms to promote robust performance. In this paper, we design artificial supervision of swarms to enable an agent to interact with a swarm of robots and command it to efficiently search complex partially known environments. This is implemented through artificial evolution of human readable behaviour trees which represent supervisory strategies. In search and rescue (SAR) problems, considering uncertainty is crucial to achieve reliable performance. Therefore, we task supervisors to explore two complex environments subject to varying blockages which greatly hinder accessibility. We demonstrate the improved performance achieved with the evolved supervisors and produce robust search solutions which adapt to the uncertain conditions.

Keywords: Swarm robotics, Artificial evolution, Behaviour trees, Search and Rescue

1 Introduction

Swarm robotics studies the application of large numbers of agents which follow simple local rules to generate complex emergent behaviours, often inspired by swarms found in nature [1]. Swarms have been applied to problems from collective motion to decentralized consensus formation [2]. Whilst swarms show great promise, in practical real-world situations we need easy ways for supervisors to change the behaviour of the swarm in an intuitive and understandable way.

Supervision of swarms has previously been investigated within the area of human swarm interaction (HSI) that includes a human operator as part of the swarm system to perform supervision [3]. Performance can be improved by monitoring the swarm's state and using different methods of interaction to fix suboptimal behaviours. Much of the work in this area has focused on methods to allow humans to interact with swarms to influence their behaviour. Many investigations employ the idea of switching between different swarm algorithms to take advantage of different emergent behaviours [4]. Other methods include interacting directly with the swarm's environment to change low level goals and manually take over control of individual agents [5]. Each control method has

shown the ability to positively affect the swarm's performance in varying scenarios. Other effects on human-swarm system performance include cognitive load on the human operator [6], their knowledge of swarm dynamics [7], and rate of interaction [8]. Our previous work has investigated methods to automate this process through the evolution of behaviour trees that can monitor and change the behaviour of the swarm. This produces human readable solutions, and enables systematic exploration of supervisory strategies [9].

In this paper, we explore the evolution of an artificial supervisor to control a swarm to search partially known, complex realistic environments. We aim to produce supervisory strategies which are robust to variations in the environment state and maintain high performance in comparison to solutions which specialize to a single environment state. The paper is structured as follows. Section 2 highlights similar areas of work and examples of exploration under uncertainty. Section 3 details the simulated scenario and the methodology used to apply artificial evolution to produce swarm supervision strategies. Section 4 then presents our findings and conclusions.

2 Related Work

Search and rescue (SAR) problems have been studied widely in swarm robotics as they are well suited for these problems through the use of large numbers of simple robots to cover large mission areas when compared to a single autonomous robot. This is demonstrated by Arnold *et al.* who show high performance with simple reactive behaviours [10]. The ability to search with limited sensing has also been demonstrated in real-world experiments with a swarm of drones exploring an indoor environment and only on-board sensing [11]. In addition, Hauert *et al.* evolve novel approaches to sweeping environments using agents with no global positioning information [12].

Examples of exploration in uncertain environments have been explored in varying levels of complexity within the field of swarm robotics. Yang *et al.* combine an ant colony search algorithm with pheromone mapping to efficiently cover an uncertain environment by limiting the amount of overlapping paths of agents [13]. Similarly, Pan *et al.* improve upon a particle-swarm optimization algorithm when searching environments in the presence of noise using optimal computing budget allocation [14]. Whilst in both cases performance is improved these problems are investigated in simple environments. Dirafzoon and Lobaton investigate the mapping of unknown environments using cockroach inspired swarm behaviours [15]. This touches on some of the concepts presented in this paper on adjusting parameters of the swarm behaviour during simulation to improve performance. Whilst the algorithms presented perform highly, this subject is discussed briefly and the environments that are investigated do not represent complex real-world environments.

3 Methodology

The following section will detail how artificial evolution has been applied to optimise swarm supervision strategies for a SAR scenario in uncertain environments. In this work we use behaviour trees (BT) to encode these supervisory strategies. We apply artificial evolution to optimize the structure of these trees and produce non-obvious high performing strategies.

3.1 Simulation

We investigate the exploration of indoor environments using a bespoke 2-D simulator. Swarm agents travel at a constant speed of *0.5m/s* and obstacle avoidance is achieved using potential fields. Agents have no perception of their surroundings beyond the avoidance of obstacles and can sense their distance to other agents. We measure the coverage of an environment based on the detection of objectives distributed over the environment which the swarm must find. An objective is detected when the euclidean distance between an agent and objective is less than *2.5m* and can only be found once.

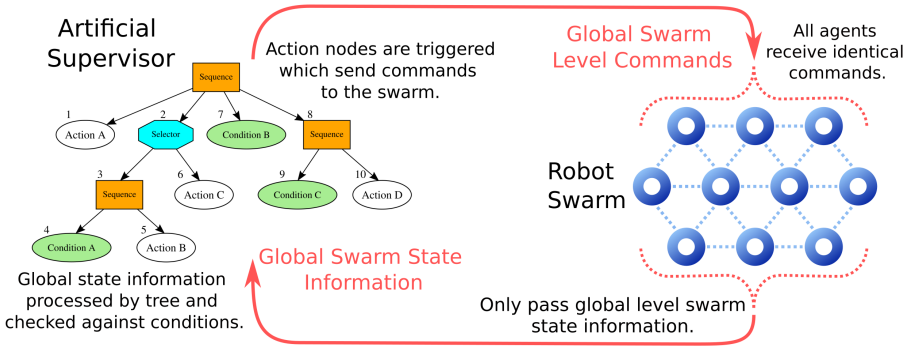
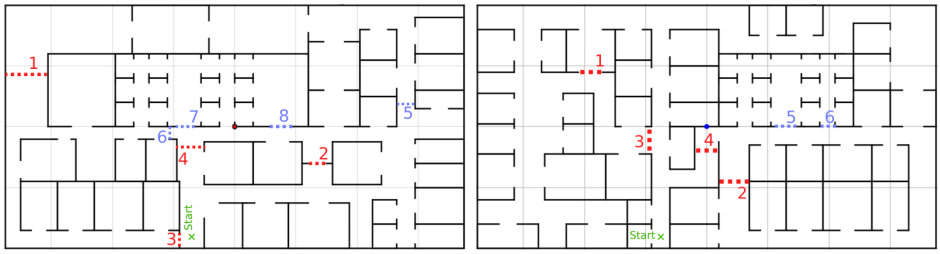


Fig. 1: Interface between the artificial supervisor and the swarm. The supervisor receives global state information from the swarm which is used to trigger different decisions. The supervisor sends global commands back to the swarm to change its behaviour.

Whilst swarm agents follow their own set of local rules, the non-embodied artificial supervisor which is separate to the swarm, observes the swarm's state at a global level and sends global level commands back to the swarm to facilitate supervision. Figure 1 presents the interaction between supervisor and swarm. The supervisor cannot perceive the shape of the environment and views the swarm through global identity-free swarm metrics without need for direct agent control. This is detailed further in section 3.3.



(a) Environment A (b) Environment B

Fig. 2: Two example environments to perform a coverage task. Shown in each are the starting positions of the swarm and numbered blockage points. The red blockages highlight those which have the most effect on navigation which are chosen for testing states. Blue blockages are excluded.

3.2 Exploring Uncertain Environment States

In this paper we investigate the problem of searching known environments with varying levels of unknown blockages that hinder navigation. To do this, we first consider where blockages might occur. In this investigation we focus on the exploration of indoor environments and study two cases shown in figure 2. Environment A, is based upon the real Bristol Robotics Laboratory floor plan and environment B is a variation of A. Both environments measure $150m \times 80m$. In each environment the swarm is deployed from the indicated starting point and has to search the environment to find objectives which are evenly distributed in increments of $2.5m$ for a total of 1800 objectives.

We explore blockages of pathways which alter the connectivity of the environment and make navigation more difficult. We first identify the worst case scenario which is the largest set of possible blockages such that the environment is not disconnected. In this case the environment can still be fully traversed, but the highest number of blockages are present. These set of blockages are highlighted in figure 2. With the blockages labelled as shown in figure 2, we measure how the addition of each individual blockage changes diffusion through the environment. Through this process we can highlight how critical blockages could change our requirements for supervisory search strategies and the need for evolution. We deploy a swarm of 1000 random walkers which are launched from the starting position and measure the probability of reaching different regions of the environment. The walkers motion is tuned to give the best level of coverage. At each position in the environment we record the proportion of time that the space is occupied over a duration of 1500s. This produces a heat map over the environment representing the probability of reaching each position.

To measure the overall difficulty of navigation, we calculate the proportion of positions with less than 1 percent probability of being occupied under each blockage. With this information we identify 4 critical blockages in each environment which reduce diffusion the most as indicated by the red blockages in figure

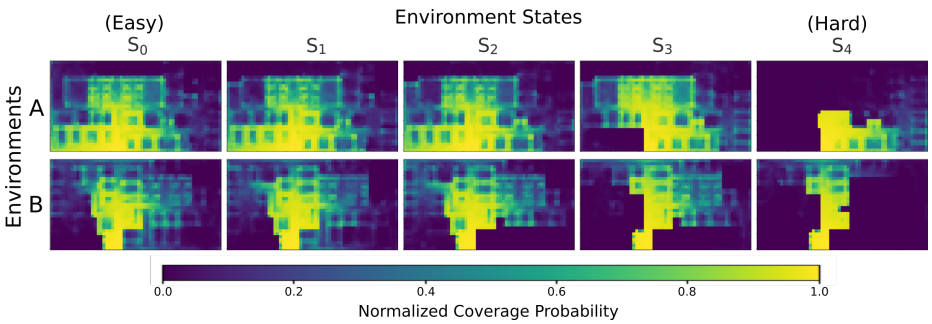


Fig. 3: Normalized probability maps of different environment states subject to blockages when deploying a swarm of 1000 random walkers. Each point in the map indicates the normalized probability that the position was covered by an agent, highlighting hard to reach blue regions.

2. The blue blockages showed lower affect and were excluded. Given the selection of these blockages, we define a set of increasingly difficult environment states to investigate.

$$s_0 = \emptyset, s_1 = \{b_1\}, s_2 = \{b_1, b_2\}, s_3 = \{b_1, b_2, b_3\}, s_4 = \{b_1, b_2, b_3, b_4\}$$

Each environment state s_0 to s_4 refers to different possible configurations of environments A and B subject to different sets of blockages. With each of these states, we observe that the proportion of the environment which is left largely uncovered increases from state s_0 to s_4 as shown in figure 3. In s_1, the addition of blockages has only slight affect on diffusion whereas s_4 greatly affects navigation. Whilst we could investigate randomly changing blockages, by targeting the most significant blockages and worst-case scenario, we reduce the need to evolve over a large set of states and more efficiently target the problem. With a set of increasingly difficult environment states to explore, we next detail the implementation of artificial supervision of swarms to robustly search these set of environment states.

3.3 Swarm Supervisor

We represent the swarm supervisor in the form of a behaviour tree (BT). In order to produce control strategies we define a set of actions and conditions that allow the supervisor to interact with the swarm and can be constructed in the form of a BT. For greater detail on this design process, refer to our previous work [9].

Methods of Interaction The supervisor can command the swarm to execute a desired swarm-level behaviour. This command is broadcast to all agents in the swarm and changes the algorithm which they execute. In addition, the supervisor can tune aspects of different behaviours by varying parameters of the local

rules. In this investigation we include the following set of search based swarm behaviours.

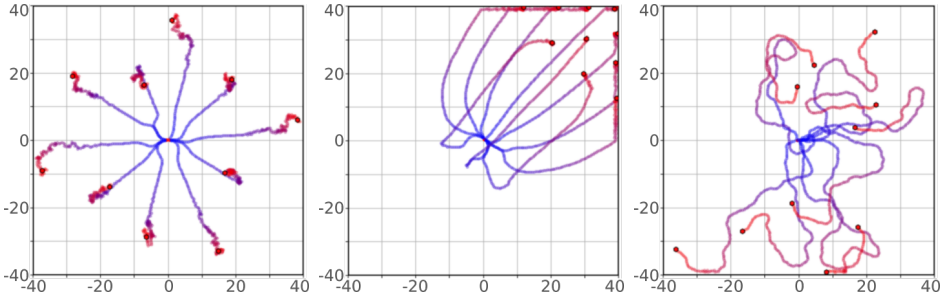
- *Dispersion*: When clustered together, agents are repelled by an exponential force from one another causing them to disperse and cover a larger area (fig 4a). When significantly spread out, agents will travel with random motion [16,17]. The supervisor can also vary a parameter, R, which scales the strength of dispersion.
- *Directed fields*: The swarm will travel in a specified direction whilst repelling nearby agents using the same rules as the dispersion behaviour (fig 4b). We enable eight varying forms of this behaviour to direct the swarm north, south, east, and west. As well as, north west, north east, south west, and south east. These behaviours give much greater control to direct the swarm to particular regions. Control over spread is also given by parameter R.
- *Random walk*: We implement a simple random walk behaviour where agents travel at a constant speed and adjust their headings each time step based on a uniform probability distribution (fig 4c). The supervisor can control the degree of random motion with parameter K.
- *Rotational random walks*: By skewing the probability that agents choose to turn in a certain direction, we generate behaviours where agents move in random looping trajectories (fig 4d). We implement two forms of this behaviour, clockwise and ant-clockwise rotation. The supervisor can also can control the turning rate with parameter J.

Conditions In order for the swarm supervisor to decide upon a certain action, knowledge of the swarm state is required. In this scenario the supervisor can observe the median position, spread of the swarm, and coverage achieved during simulation. The *median position* of the swarm is given in both the x direction, μ_x, and y direction, μ_y. *Spread*, σ, is defined as the average distance from agent to agent as defined in equation 1, where n is the total number of agents and each agent ordinate is defined as x_n and y_n. Coverage, γ, is the proportion of detected objectives to the total number of objectives.

This high level representation means the supervisor does not use knowledge of each agent state to enable control. In addition, the supervisor has no knowledge of the structure of the environment and must learn this through the evolutionary process.

$$\sigma = \frac{1}{n(n-1)} \sum_{k=1}^{n} \sum_{i=1:i\neq k}^{n} \sqrt{(x_i - x_k)^2 - (y_i - y_k)^2)} \tag{1}$$

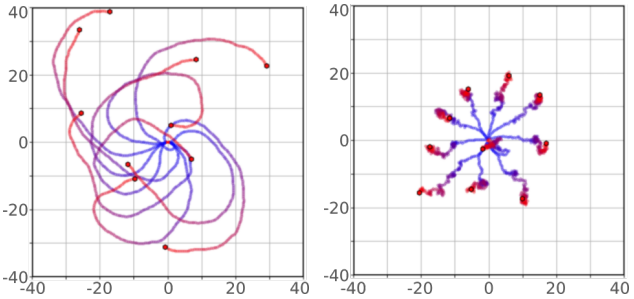
Condition nodes are constructed using these real-time metrics when compared to defined thresholds shown in table 1. This enables the supervisor to perform simple checks to see if a metric is greater or less than a set value. The thresholds that can be selected for the median are bounded within the size of the environment, and similarly, the spread is limited to a high level of dispersion within the bounds of the environment.



(a) Dispersion with high spread.

(b) Northeast directed field.

(c) Random walk.



(d) Clockwise random walk.

(e) Dispersion with reduced spread.

Fig. 4: Examples of available swarm behaviours used for navigation. Each figure depicts the swarm motion over 150 time steps with agent positions shown with shifting colour from blue to red with the progression of time.

3.4 Evolving the Swarm Supervisor

We apply genetic programming (GP) to evolve BTs and optimize their structure in order to produce high performing supervisory strategies. GP has previously been applied to BTs in other applications [18]. Table 1 presents the available nodes used to construct trees and highlights the limits that the conditional statements must satisfy.

Evolutionary Algorithm We evaluate the fitness of strategies by the proportion of objectives that are detected. The reward for finding each objective also decays over time in order to promote fast exploration. Each objective has a unique decay constant based on the probability maps discussed in section 3.2. The probability of finding each objective scales the rate of decay, ρ, by equation 2 where δ is the probability of finding an objective, t is the current time, and T is the total time duration. Therefore, objectives with a low probability of being found decay slower, maintaining a higher reward. Each environment state under investigation has a unique probability map such that the rate of decay varies

Table 1: The limits defined by the GP algorithm for the types of nodes that can be selected to produce BTs.

Node type	Selection Choices
Operator	Selector / Sequence (Between 2-7 children)
Action node	Emergent behaviours: Dispersion / North / South / East / West / North East / North West / South East / South West / Random walk / Clockwise random walk / Anti-clockwise random walk Param set: $R \in$ [1, 10, 20, 30, 40, 50, 60] $J \in$ [0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09] $K \in$ [0.005, 0.01, 0.015, 0.02, 0.025, 0.03, 0.035, 0.04]
Condition node	μ_x>-38, μ_x>-34, μ_x>-30, ... increment by 4 ..., μ_x>30, μ_x>34, μ_x>38 μ_x<-38, μ_x<-34, μ_x<-30, ... increment by 4 ..., μ_x<30, μ_x<34, μ_x<38 μ_y>-38, μ_y>-34, μ_y>-30, ... increment by 4 ..., μ_x>30, μ_y>34, μ_y>38 μ_y<-38, μ_y<-34, μ_y<-30, ... increment by 4 ..., μ_x<30, μ_y<34, μ_y<38 σ>2, σ>6, σ>10, ... increment by 4 ..., σ>26, σ>30, σ>34 σ<2, σ<6, σ<10, ... increment by 4 ..., σ<26, σ<30, σ<34 γ>0.1, γ>0.2, γ>0.3, ... increment by 0.1 ..., γ>0.7,γ>0.8, γ>0.9 γ<0.1, γ<0.2, γ<0.3, ... increment by 0.1 ..., γ<0.7,γ<0.8, γ<0.9

depending on the difficulty of each state. This helps scale the fitness reward in accordance with the difficulty of each environment state.

$$\rho = \exp\left(-3\frac{\delta t}{T}\right) \tag{2}$$

The fitness function used to evaluate solutions is then defined by equation 3. Each individual has n attempts to search the environment which is set to *4*. The score per run, β, is the summation of all objectives detected and their associated rewards and, α, is the total number of objectives. The size of evolved BTs are also limited to a maximum number of nodes τ. This applies pressure on the evolution to find concise and efficient solutions. When training a supervisor over different environment states, the fitness is averaged across each of those states. Evolution aims to maximize fitness.

$$Fitness = \begin{cases} \dfrac{1}{n\alpha}\displaystyle\sum_{k=1}^{n}\beta & \text{if } \tau <= 100 \\ 0 & \text{if } \tau > 100 \end{cases} \tag{3}$$

We implement a standard evolutionary island model using three islands with identical conditions [19]. We ran evolutionary runs over 300 generations with a population of *40* individuals on each island. At each generation, we used tournament selection with groups of *3* individuals followed by single point crossover, single point mutation, and sub-tree growth. Elitism is used to save the best *6* individuals from each generation. For each search, we set a time limit of *1500* seconds. This time period is sufficiently long to fully explore the environments.

4 Results

4.1 Behaviour Benchmarks

Before evaluating the performance of the evolved supervisory strategies, we test the performance of each individual swarm behaviour when deployed without supervision in each environment state (fig 5a and 5b) to highlight where evolved supervision is needed.

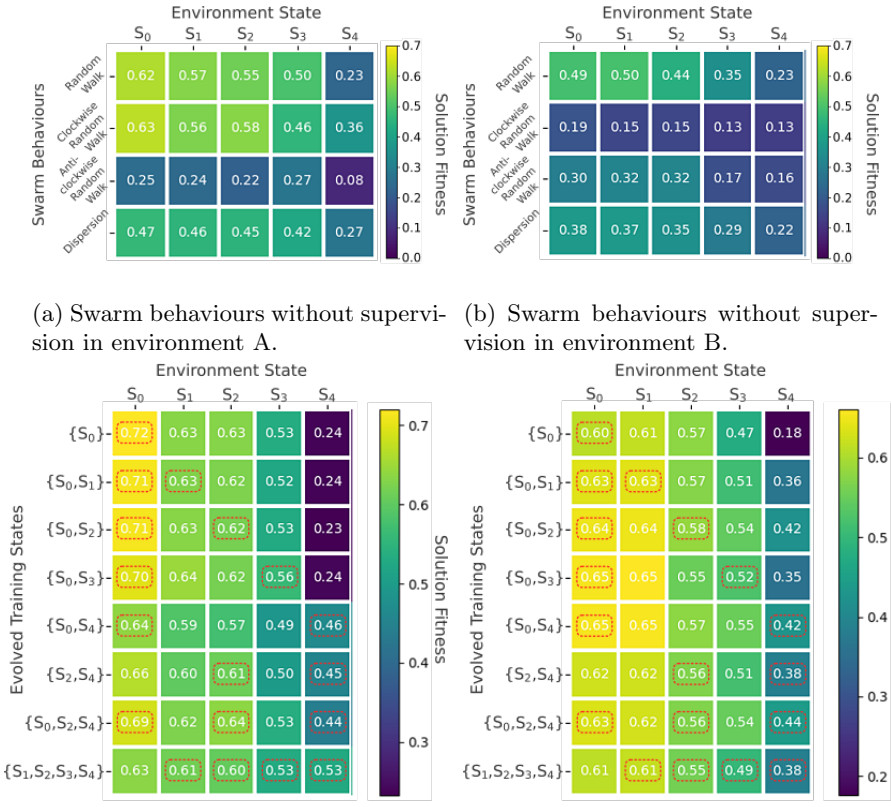
For environment A, we see that the random behaviours perform well, with the clockwise walk performing the highest whilst the random walk and dispersion also perform well. When deploying this behaviour we see that it is very useful for cycling in and out of rooms, following the edge of the interior, before then exiting to continue through the environment (fig 6a). For environment B, we see that again dispersion performs well, whereas, the rotational walks perform much lower than in environment A. In this case, there are several places where agents can become stuck looping over the same area when passing through certain pathways (fig 6b). The use of the rotation behaviours appear very effective in certain types of environments where many paths are open, however in environment B, these types of motions are not suitable in bent corridors. The benchmarks highlight that we cannot rely on these singular behaviours to solve these problems and highlight that behaviours become susceptible to changes in environment states, causing significant drops in performance.

4.2 Evolved Supervisory Control

The aim of this investigation is to produce solutions that can robustly search environments under the presence of unknown blockages. We first evolve supervision of the swarm in only state s_0 where no blockages are present and the supervisor is not exposed to changes in the environment state. We then run further configurations where each supervisor is exposed to at least two possible environment states during evolution. These different training configurations are presented in figures 5c and 5d. Each row shows the performance of an evolved supervisor which has been subject to a unique set of training states during evolution. We see the increase in performance in relation to figures 5a and 5b demonstrates the value of artificial supervision over the deployment of an unsupervised swarm. In addition, we observe the ability to produce generalizable solutions where supervisors achieve high performance across each environment state. In the following sections we examine these solutions and how they form robust strategies.

4.3 Qualitative Analysis of Evolved Supervision

In both environments, supervisors interacted with the swarm by selecting specific sequences of swarm behaviours with fine tuned control of spread and degree of random motion. We found that supervisors opted to use combinations of the random walk, rotational walks, and the directed field behaviours. We saw very few uses of dispersion despite performing well in the benchmarks.



(a) Swarm behaviours without supervision in environment A.

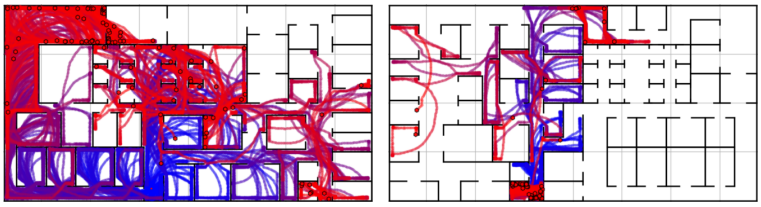
(b) Swarm behaviours without supervision in environment B.

(c) Evolved solutions in environment A.

(d) Evolved solutions in environment B.

Fig. 5: Fitness of the evolved supervisory strategies under each possible environment state. Each data point is the average performance over 100 trials. Each row represents the best evolved supervisor from each set of training states. The red boxes indicate the performance in the states of which the supervisor was trained. Supervisors trained over multiple environment states produce more robust performance.

If we first examine the supervisor trained in only state s_0 in environment A, we see that the supervisor scores highly when searching in state s_0 as expected. The supervisor initially uses the random walk behaviour to spread out in all directions before switching to the clockwise walk. The random walk performs well initially, however agents become trapped in rooms more easily after a short period of time. The supervisor identifies this and switches to the clockwise random walk which is better suited to enter and exit rooms. The supervisor finishes by commanding the swarm to head east, forming a final sweep of the environment near the end of the search period. This approach was able to achieve up to 94



(a) Clockwise random walk coverage in environment A

(b) Clockwise random walk coverage in environment B

Fig. 6: Coverage of a swarm of 100 agents when performing only a clockwise random walk. Agent trajectories are plotted with colour shifting from blue to red over time. Whilst very effective in environment A, the behaviour breaks easily under the right conditions where agents become trapped in environment B.

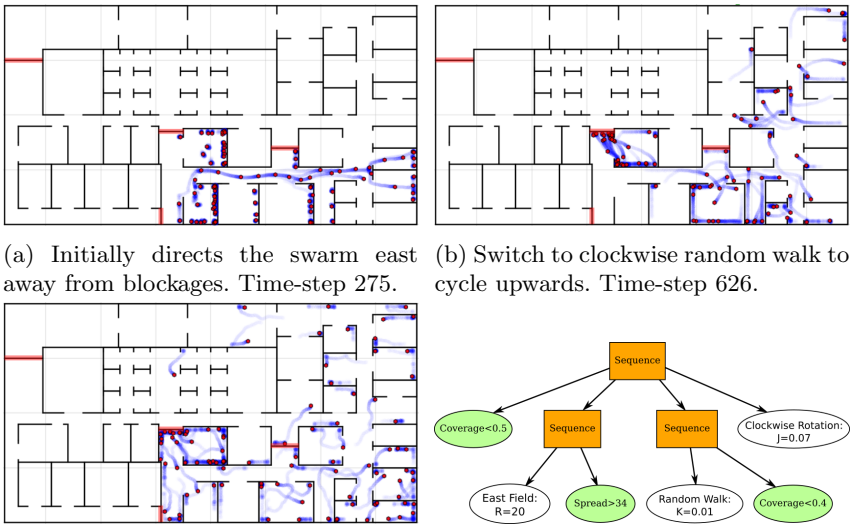
percent coverage of the environment. This shows the benefit of the supervisor to identify the advantages of each type of behaviour and combine them into an effective search strategy.

Whilst this supervisor performs well in state s_0, when evaluated in the blocked states we see that performance drops quickly with very low performance in s_4 as shown in fig 5c. Under the additional blockages the supervisor attempts to use the random walk which causes the majority of agents to become stuck near the starting position resulting in minimal exploration.

4.4 Robust Search Strategies

In environment A, when training over s_0 and one additional state up to s_3, there is shown to be little variation in overall performance and still poor performance in s_4. This suggests that the addition of blockages s_1 to s_3 do not have significant effect on how we should change supervision.

However, when the supervisor is evolved over s_4, we see that performance in this state is greatly improved. Every supervisor trained including state 4 takes a different approach to the previous supervisors as illustrated in fig 7 and shown in simulation (video link). Rather than first using random behaviours, 3 of the solutions first direct the swarm east away from the additional blockage stopping access to the center of the environment (fig 7a). This directs the swarm across to the bottom right of the environment to avoid the blockages. In one case the north movement is used with high spread which inadvertently pushes agents east then up the right side of the environment. Under state s_4, using the random behaviours initially performs poorly where the use of the clockwise random walk causes agents to become trapped within a loop near the starting point. This is the reason why solutions on row 1 to 4 which aren't trained over s_4 perform poorly. After being directed around the blockages in s_4, the supervisors switch to the clockwise random walk which becomes effective at passing upwards



(a) Initially directs the swarm east away from blockages. Time-step 275.

(b) Switch to clockwise random walk to cycle upwards. Time-step 626.

(c) Finish by switching to random walk for greater coverage. Time-step 1420.

(d) The supervisory behaviour tree evolved over states s_1 to S_4.

Fig. 7: The behaviour of the supervisor trained over states s_1 to s_4 when deployed in environment A. The stages of this solution are captured when searching the environment in state s_4 showing the swarm being diverted around blockages highlighted in red.

and around the blockages (fig 7b) and finishing with a normal random walk (fig 7c). In addition, when these solutions are deployed in states s_0 to s_3 with fewer blockages, this approach is still effective and the supervisor switches faster to the clockwise random walk behaviour, identifying the lack of blockages.

4.5 Performance Variation Between Environments

In environment B, we observe different trends. As previously highlighted, we see a lack of robustness for the supervisor trained only in s_0. This is because the supervisor learns to use the random clockwise walk initially which performs poorly under the presence of additional blockages, causing agents to become trapped in certain areas. However, when trained over one additional environment state with s_1, we see improved performance even in s_4 despite not being trained over that state. This suggests that in this case, it's not crucial to expose the supervisor to the most difficult state s_4.

We also notice that the supervisor trained on s_0 and s_3 performs highly in each environment state but also achieves the highest scores in s_0 and s_1, outperforming the supervisor which specialises to s_0. These findings suggest in this case that it's always better to give the supervisor a small amount of exposure to other environment states to promote higher performance in all states without having to compromise performance in s_0. In both environments we observe

that given exposure to additional states of the environment, the supervisor is capable of adapting its strategy to maintain good performance in the worst case scenario. Through this exposure, the supervisors identify where the most significant blockages occur and actively directs the swarm away from these areas of the environment, taking the safest route and ensuring robust performance.

5 Conclusions and Future Work

In order to achieve real-world deployment of swarms, we need to consider how to effectively monitor and influence their behaviour. The implementation of supervisory control of swarms enables us to promote high performing, robust solutions to complex problems with only minimal global level interaction. In this paper we showed that through the application of artificial swarm supervision, we were able to take advantage of different swarm behaviours to effectively search complex environments. We identified that when training a supervisor without the presence of uncertainty, the emergent strategies were not robust to variations in the environment. By evolving over the worst case scenario where critical blockages were present, we observed that the supervisor learned to direct the swarm away from these blockage points and adapt its approach and use different behaviours. We also identified that evolving over only slight variations in the environment made supervision significantly more robust to the most critical blockages which it had not been exposed to. All of these supervisory solutions were achieved using concise sequences of behaviours, relying on the autonomy of the swarm and learning how best to utilise the available behaviours.

Future work will explore how we can evolve robust supervision with the worst case scenario in greater detail and how this compares to evolving over randomly chosen states. In addition, we will investigate a broader spectrum of environments to understand to what extent our findings generalize.

6 Acknowledgements

This work was funded and delivered in partnership between the Thales Group and the University of Bristol, and with the support of the UK Engineering and Physical Sciences Research Council Grant Award EP/R004757/1 entitled "Thales-Bristol Partnership in Hybrid Autonomous Systems Engineering (T-B PHASE)".

References

1. X. B. Meng, X. Z. Gao, L. Lu, Y. Liu, and H. Zhang, "A new bio-inspired optimisation algorithm: Bird Swarm Algorithm," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 28, no. 4, pp. 673–687, 2016.
2. G. Valentini, E. Ferrante, and M. Dorigo, "The best-of-n problem in robot swarms: Formalization, state of the art, and novel perspectives," *Frontiers Robotics AI*, vol. 4, no. MAR, 2017.

3. A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, "Human Interaction with Robot Swarms: A Survey," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 9–26, 2016.
4. A. Kolling, S. Nunnally, and M. Lewis, "Towards human control of robot swarms," *HRI'12 - Proceedings of the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction*, pp. 89–96, 2012.
5. P. Walker, S. A. Amraii, M. Lewis, N. Chakraborty, and K. Sycara, "Control of swarms with multiple leader agents," *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pp. 3567–3572, 2014.
6. A. Kolling, K. Sycara, S. Nunnally, and M. Lewis, "Human Swarm Interaction: An Experimental Study of Two Types of Interaction with Foraging Swarms," *Journal of Human-Robot Interaction*, vol. 2, pp. 104–129, 6 2013.
7. G. Kapellmann-Zafra, N. Salomons, A. Kolling, and R. Groß, "Human-robot swarm interaction with limited situational awareness," in *International Conference on Swarm Intelligence*, vol. 9882 LNCS, pp. 125–136, Springer, 2016.
8. P. Walker, S. Nunnally, M. Lewis, N. Chakraborty, and K. Sycara, "Levels of automation for human influence of robot swarms," *Proceedings of the Human Factors and Ergonomics Society*, pp. 429–433, 2013.
9. E. Hogg, S. Hauert, D. Harvey, and A. Richards, "Evolving behaviour trees for supervisory control of robot swarms," *Artificial Life and Robotics*, vol. 25, no. 4, pp. 569–577, 2020.
10. R. D. Arnold, H. Yamaguchi, and T. Tanaka, "Search and rescue with autonomous flying robots through behavior-based cooperative intelligence," *Journal of International Humanitarian Action*, vol. 4, no. 1, 2019.
11. T. Stirling, J. Roberts, J. C. Zufferey, and D. Floreano, "Indoor navigation with a swarm of flying robots," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4641–4647, 2012.
12. S. Hauert, J. C. Zufferey, and D. Floreano, "Evolved swarming without positioning information: An application in aerial communication relay," *Autonomous Robots*, vol. 26, no. 1, pp. 21–32, 2009.
13. F. Yang, X. Ji, C. Yang, J. Li, and B. Li, "Cooperative Search of UAV Swarm Based on Improved Ant Colony Algorithm in Uncertain Environment,"
14. H. Pan, L. Wang, and B. Liu, "Particle swarm optimization for function optimization in noisy environment," *Applied Mathematics and Computation*, vol. 181, no. 2, pp. 908–919, 2006.
15. A. Dirafzoon and E. Lobaton, "Topological mapping of unknown environments using an unlocalized robotic swarm," *IEEE International Conference on Intelligent Robots and Systems*, pp. 5545–5551, 2013.
16. T. R. Hsiang, E. M. Arkin, M. A. Bender, S. P. Fekete, and J. S. Mitchell, "Algorithms for rapidly dispersing robot swarms in unknown environments," *In Algorithmic Foundations of Robotics V*, pp. 77–93, 2004.
17. J. McLurkin and J. Smith, "Distributed Algorithms for Dispersion in Indoor Environments Using a Swarm of Autonomous Mobile Robots," *Distributed Autonomous Robotic Systems 6*, pp. 399–408, 2008.
18. S. Jones, M. Studley, S. Hauert, and A. Winfield, "Evolving Behaviour Trees for Swarm Robotics," *Springer Tracts in Advanced Robotics*, pp. 487–501, 2018.
19. G. Squillero and A. Tonda, "Divergence of character and premature convergence: A survey of methodologies for promoting diversity in evolutionary optimization," *Information Sciences*, vol. 329, pp. 782–799, 2016.